

ALGORITMO DE RICART Y AGRAWALA: EXCLUSIÓN MUTUA DISTRIBUIDA

2025

ALGORITMO

- Coordenação entre N processos para conceder entrada em uma determinada região crítica
- Região crítica é o acesso para um recurso compartilhado que não deve ser acessado por mais de um processo ou thread em execução.



ALGORITMO

- Os processos que desejam entrar em uma seção crítica devem ter a aprovação de todos os outros nós envolvidos na coordenação.
- Sem a aprovação de todos os outros, um nó não pode acessar essa seção crítica.





ALGORITMO

- 
1. REQUEST: Quando um processo quer entrar na seção crítica, envia uma mensagem REQUEST (com seu timestamp) a todos os outros.

ALGORITMO

- 1. REQUEST: Quando um processo quer entrar na seção crítica, envia uma mensagem REQUEST (com seu timestamp) a todos os outros.
- 2. REPLY ou DEFER:
 - Se o receptor não está interessado na seção crítica ou tem um timestamp maior, responde imediatamente com REPLY.
 - Se o receptor também quer acessar e tem um timestamp menor, defer (só responde depois de liberar a seção crítica).



ALGORITMO

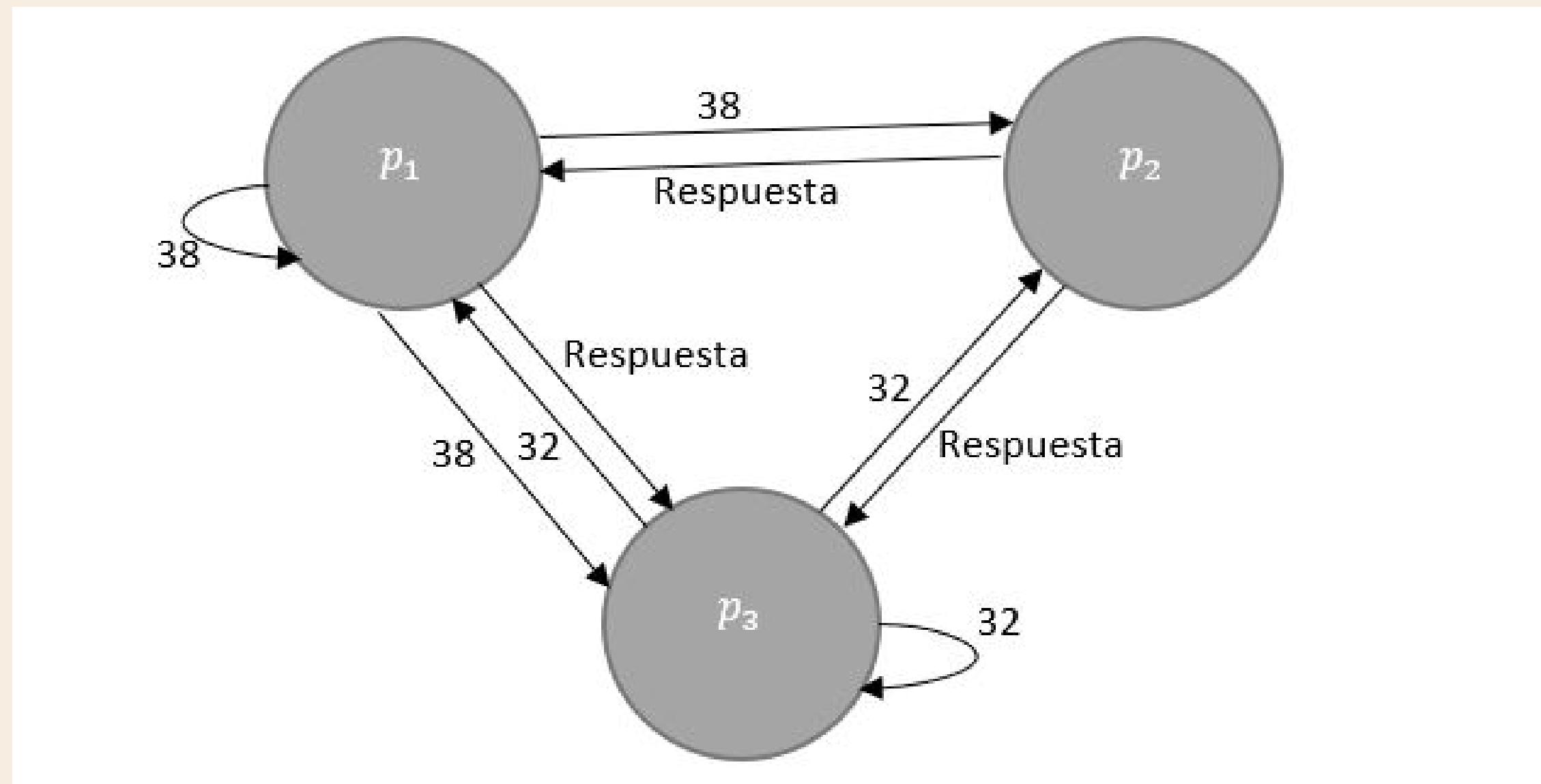
- 1. REQUEST: Quando um processo quer entrar na seção crítica, envia uma mensagem REQUEST (com seu timestamp) a todos os outros.
- 2. REPLY ou DEFER:
 - Se o receptor não está interessado na seção crítica ou tem um timestamp maior, responde imediatamente com REPLY.
 - Se o receptor também quer acessar e tem um timestamp menor, defer (só responde depois de liberar a seção crítica).
- 3. ENTER: O processo entra na seção crítica quando recebe REPLY de todos.



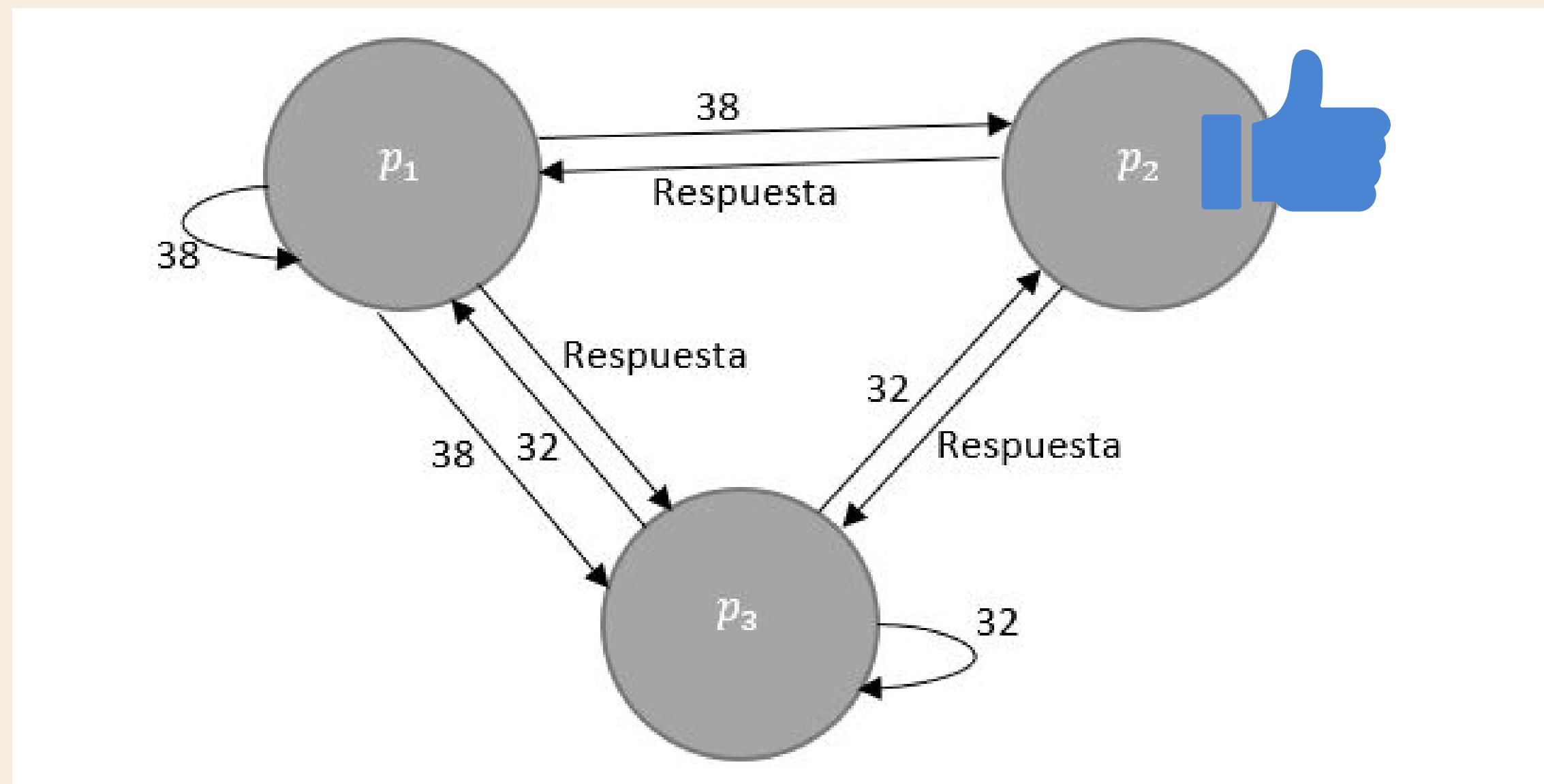
ALGORITMO

- 1. REQUEST: Quando um processo quer entrar na seção crítica, envia uma mensagem REQUEST (com seu timestamp) a todos os outros.
 - 2. REPLY ou DEFER:
 - Se o receptor não está interessado na seção crítica ou tem um timestamp maior, responde imediatamente com REPLY.
 - Se o receptor também quer acessar e tem um timestamp menor, defer (só responde depois de liberar a seção crítica).
 - 3. ENTER: O processo entra na seção crítica quando recebe REPLY de todos.
 - 4. RELEASE: Ao sair, envia RELEASE aos processos que deferiram, liberando seus REPLYs.
- 

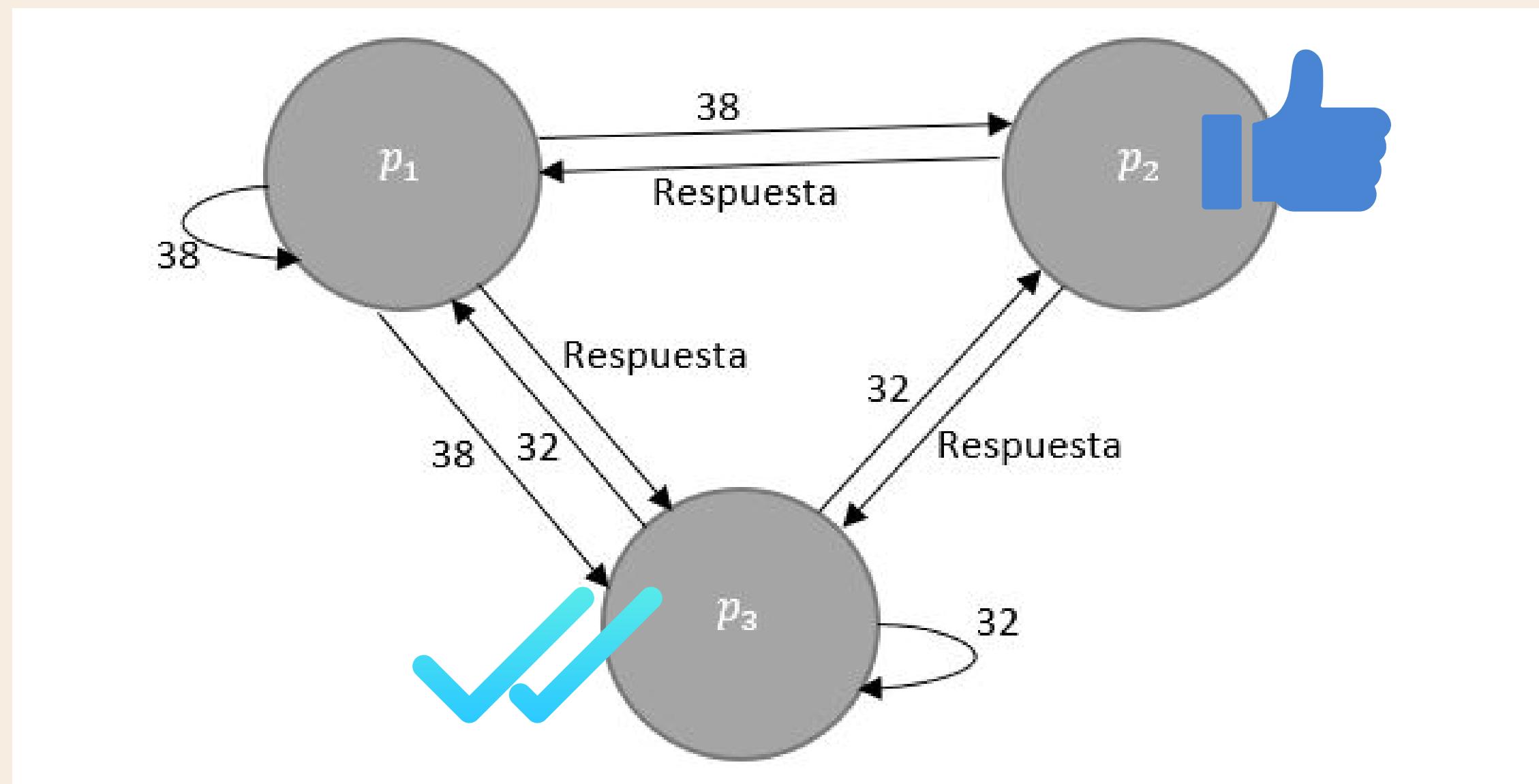
EXEMPLO



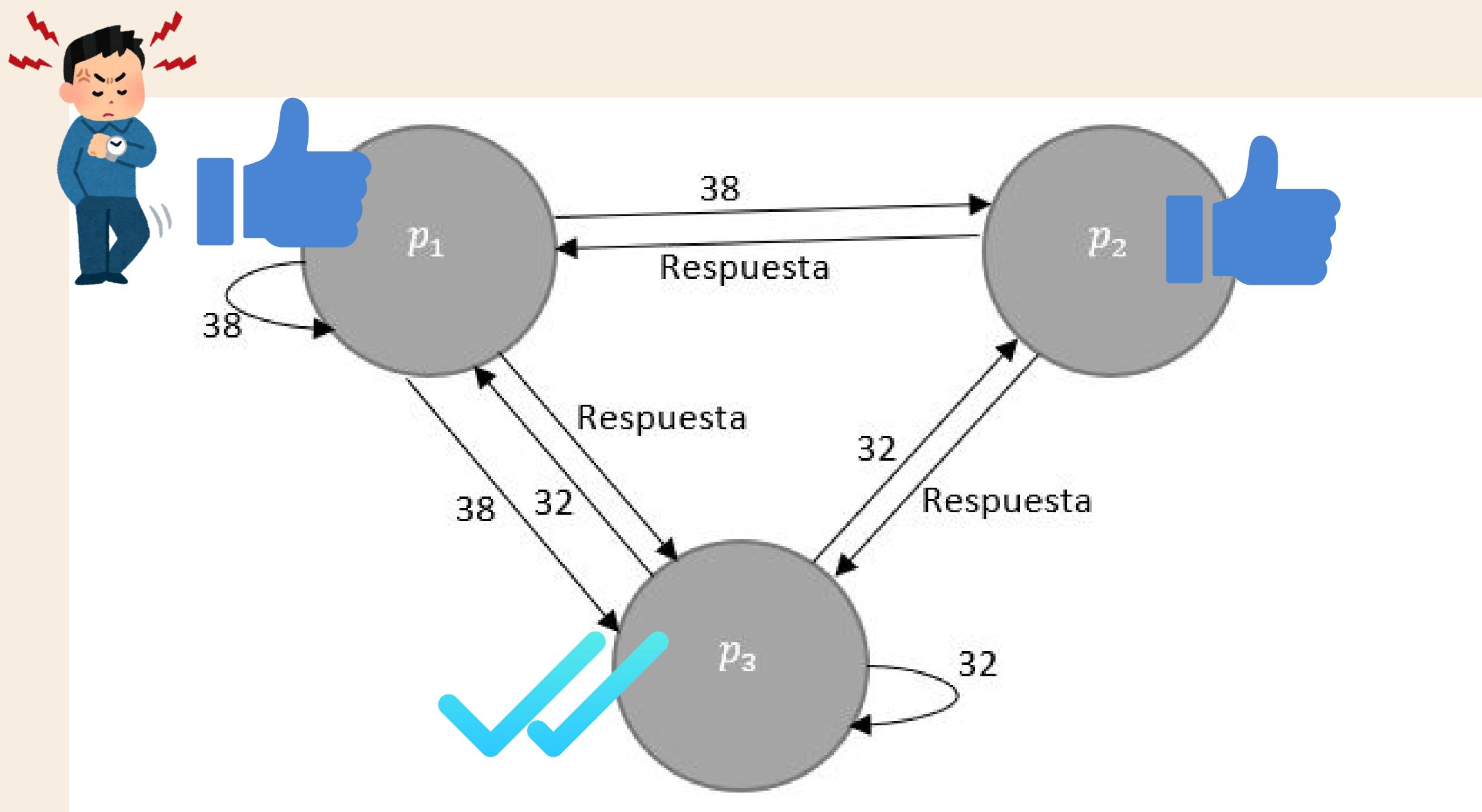
EXEMPLO



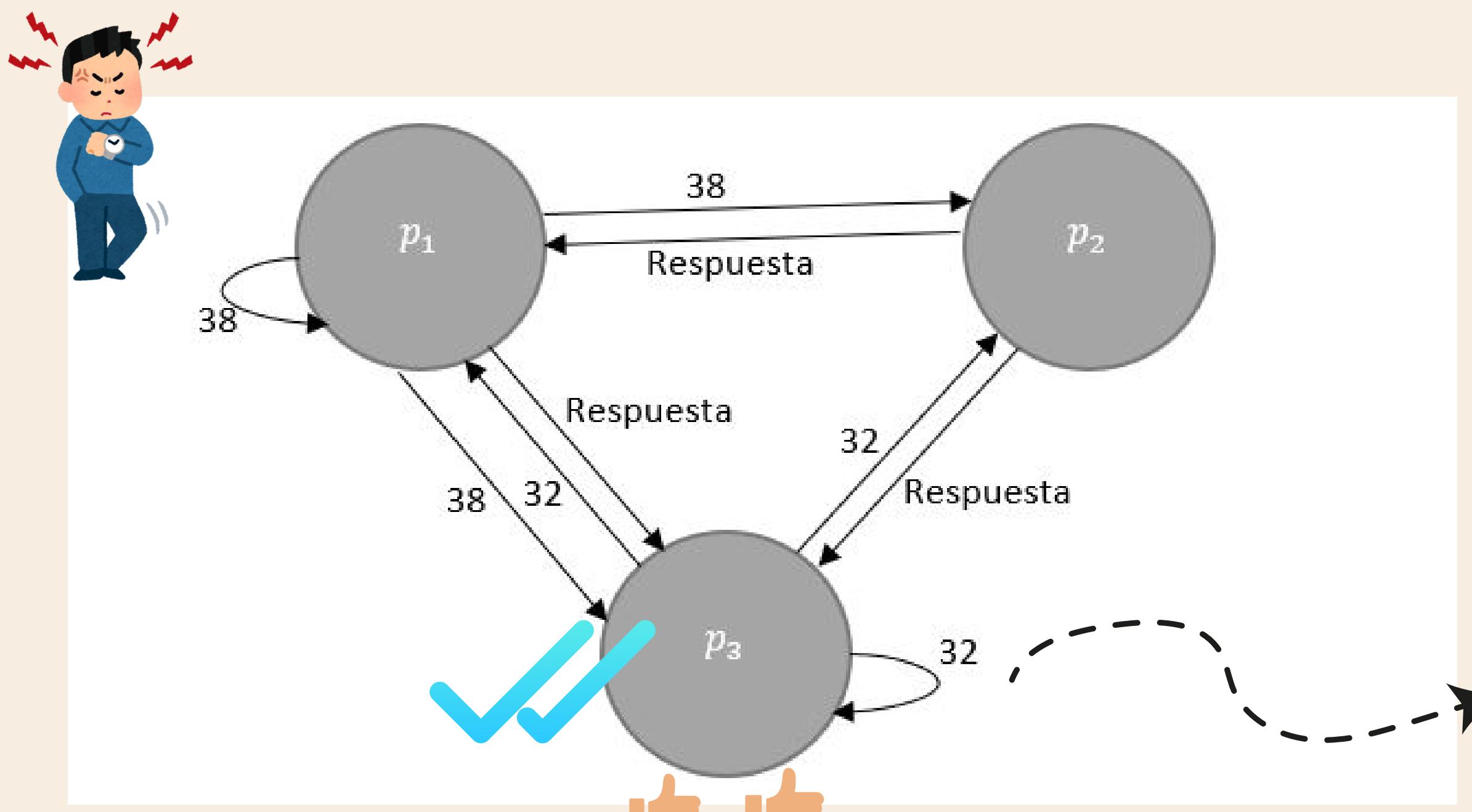
EXEMPLO



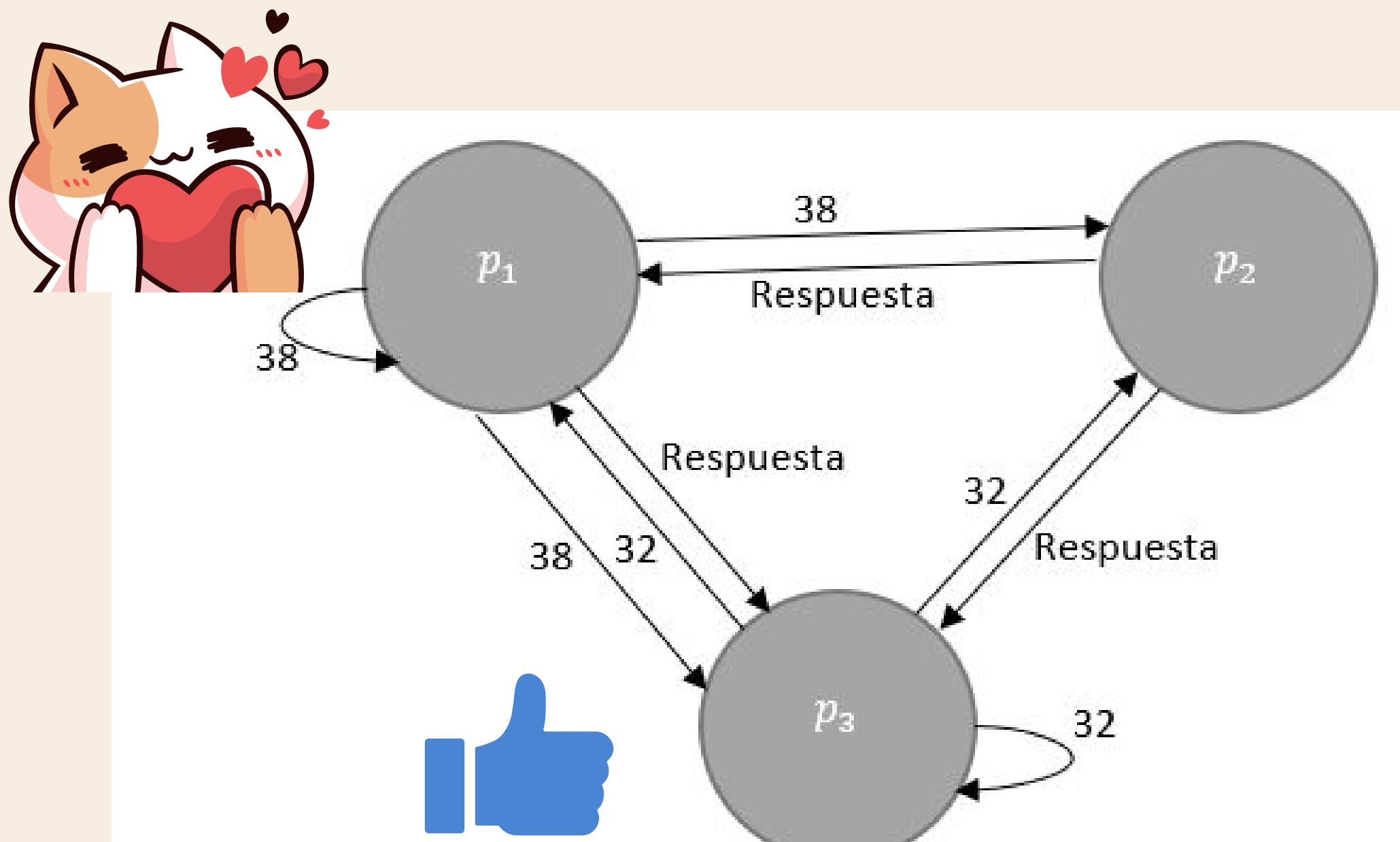
EXEMPLO



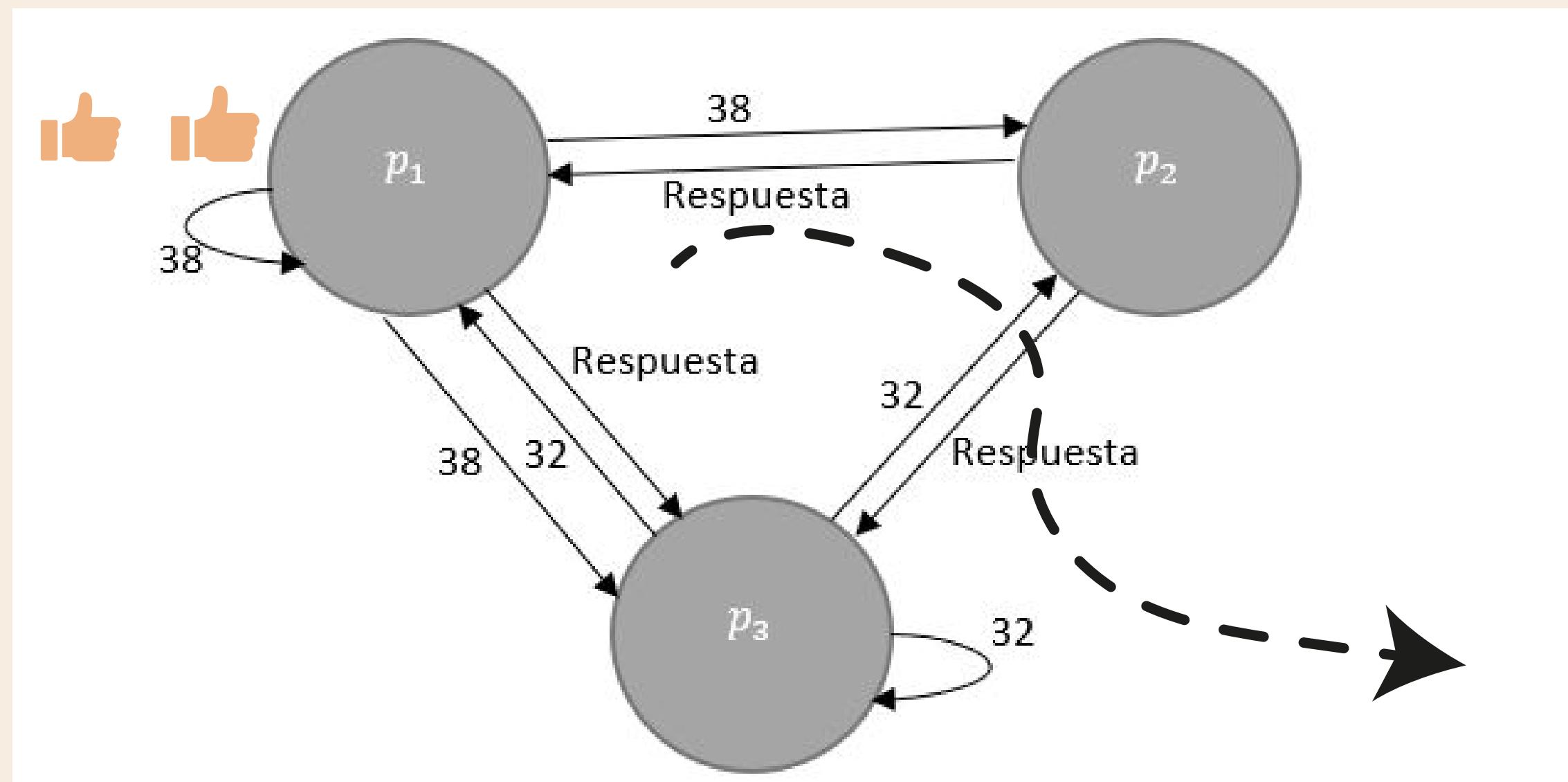
EXEMPLO



EXEMPLO



EXEMPLO



Presumimos que p2 não esteja interessado em entrar na região crítica, enquanto **p3 e p1 estão e a solicitam com as mensagens <32,p3> e <38,p1>**, respectivamente. Ao comparar os carimbos de data e hora de cada uma das mensagens, **a solicitação de p1 é maior que a de p3**. Quando p2 recebe as solicitações de ambos, como não está interessado em entrar na seção crítica, ele responde imediatamente a ambas. *Quando p3 recebe a solicitação de p1, ele compara seu registro de data e hora, que é 32, com o da mensagem recebida, que é 38, e como seu registro de data e hora é menor que o de p1, ele não responde, mantendo p1 sem permissão para entrar na região crítica.* Da mesma forma, quando p1 recebe a mensagem de p3 e compara os registros de data e hora, tendo um registro de data e hora maior que o de p3, ele responde imediatamente. Quando p3 recebe as N-1 respostas, ele pode entrar na seção crítica. Quando p3 sair dela, ele responderá a p1 de forma a garantir sua entrada.



DESVANTAGENS

- Requer comunicação com todos os processos (não escalável para muitos nós).
 - Pode ter deadlock se mensagens forem perdidas ou se algum nó falha, pois os nós esperam receber $N-1$ mensagens para entrar na região crítica.
- 

DESVANTAGENS

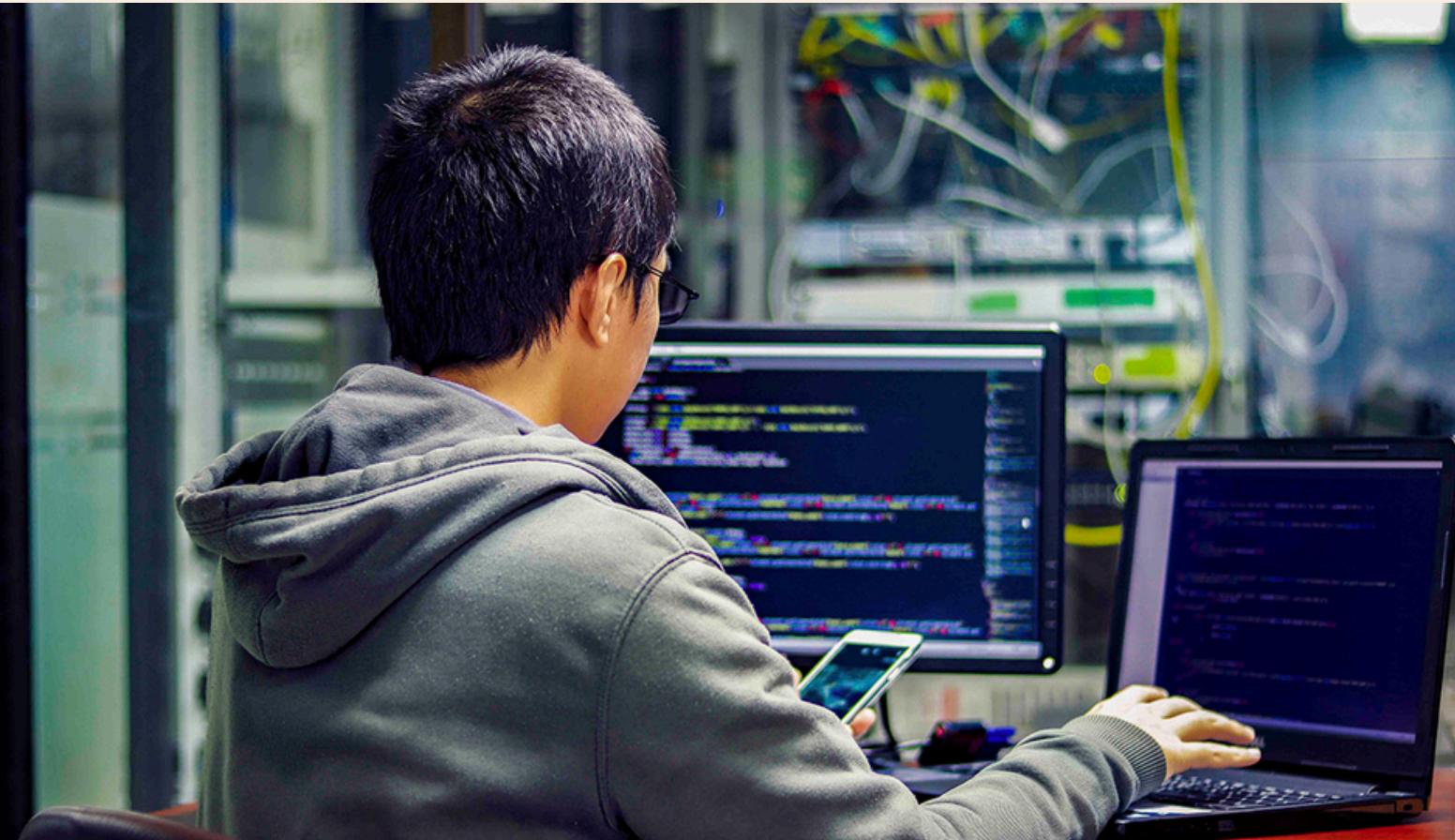
- Requer comunicação com todos os processos (não escalável para muitos nós).
- Pode ter deadlock se mensagens forem perdidas ou se algum nó falha, pois os nós esperam receber $N-1$ mensagens para entrar na região crítica.





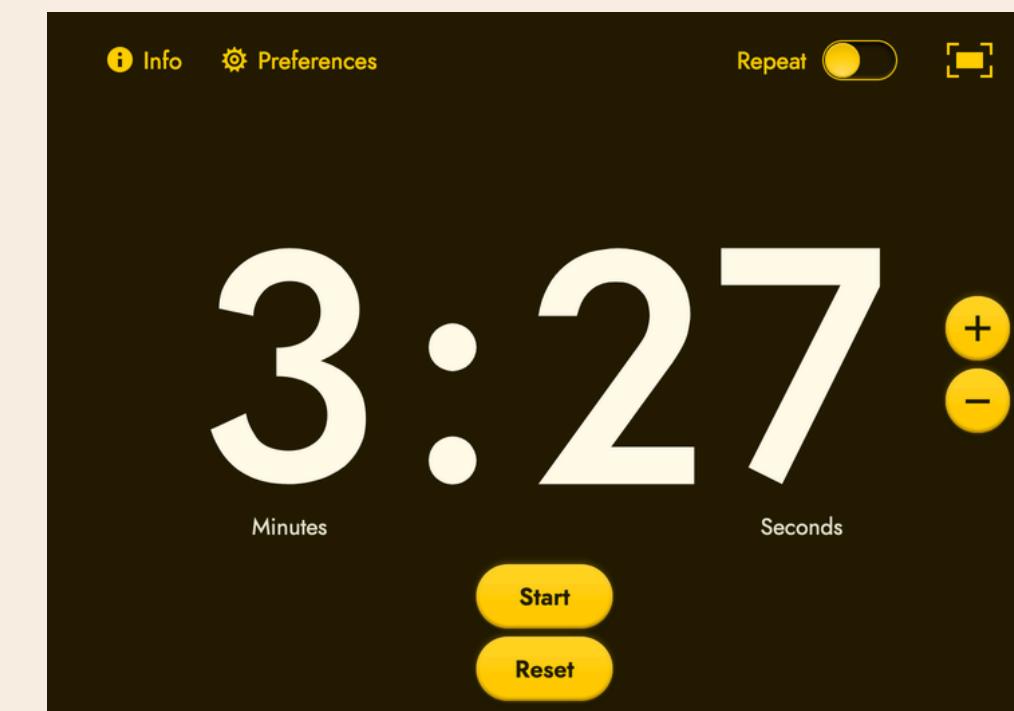
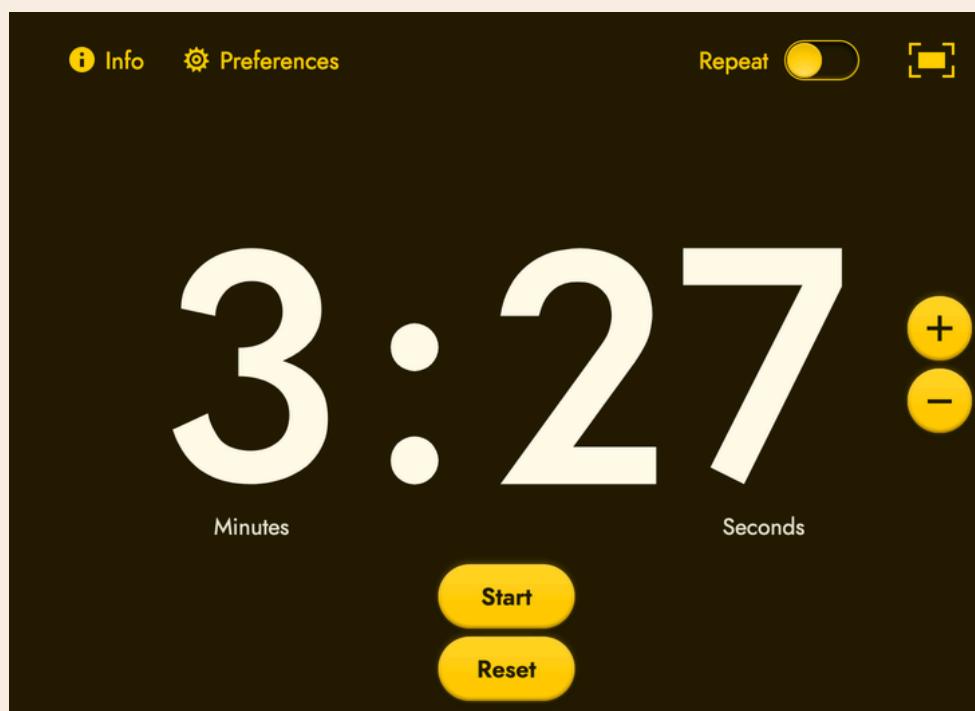
VANTAGENS

- Não requer um coordenador centralizado
- Justo? (ordem por timestamps)



VANTAGENS

→ Ordem por timestamps (Criterio de Desempate)



VANTAGENS

→ Ordem por timestamps (Timestamps Incorretos)



VAMOS
ESCLARECER
SUAS DÚVIDAS!

2025

